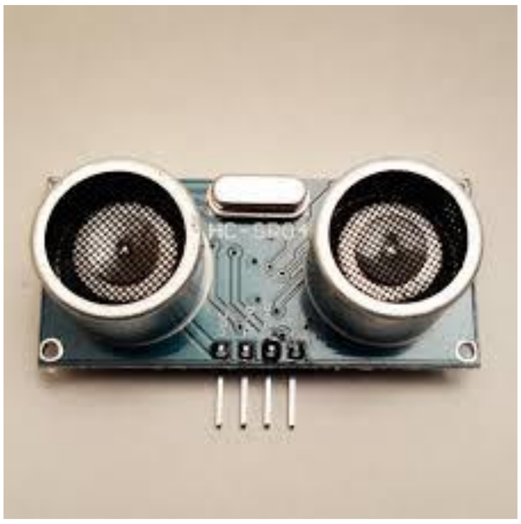
HCSR04 Ultrasonic sensor and STM32

Ultrasonic ranging module HC – SR04 provides a 2cm – 400cm non-contact measurement function, and the ranging accuracy can reach 3mm.

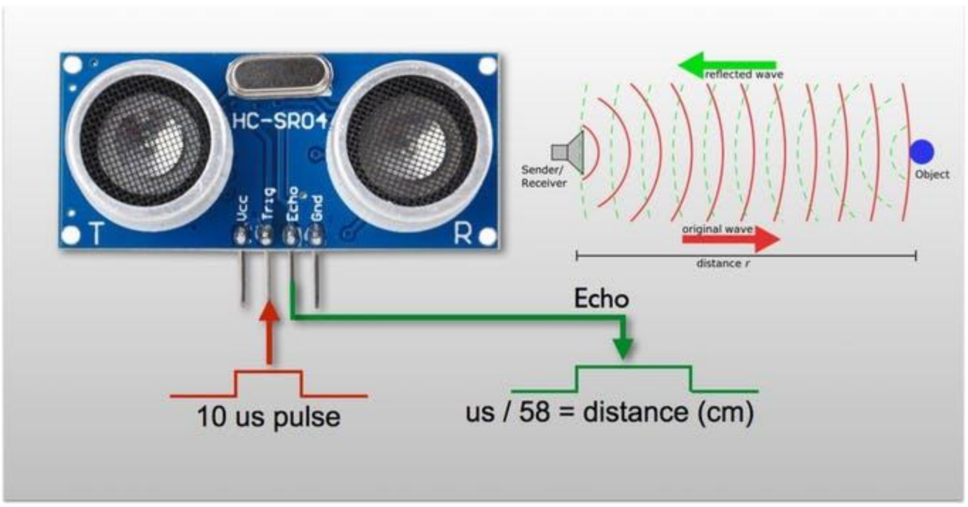
The modules include ultrasonic transmitters, receivers, and control circuits.

Today in this tutorial we are going to learn How to interface the HCSR04 Ultrasonic sensor module with STM32.



- Working of hcsr04 is pretty simple and straight.
- The module emits an ultrasound at 40 KHz which, after reflecting from obstacle, bounces back to the module.
- By using the travel time and the speed of the sound, we can calculate the distance between the sender and the obstacle.

According to the datasheet of hc-sr04, the following is required to be done :-



- Keep the Trig pin HIGH for at least 10us
- The Module will now send 8 cycle burst of ultrasound at 40 kHz and detect whether there is a pulse signal back
- IF the signal returns, module will output a HIGH PULSE whose width will be proportional to the range of the object.
- Distance can be calculated by using the following formula **:- range = high level time * velocity** (340m/s) / 2
- We can also use **uS / 58 = Distance in cm or uS / 148 = distance in inch**
- It is recommended to wait for at least 60ms before starting the operation again.

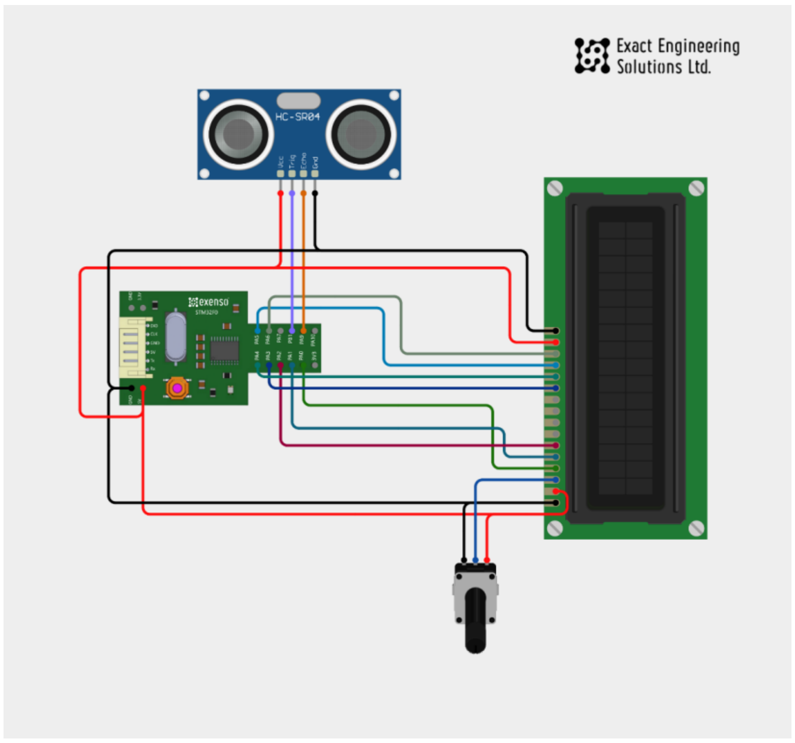
Components Required

You will need the following components –

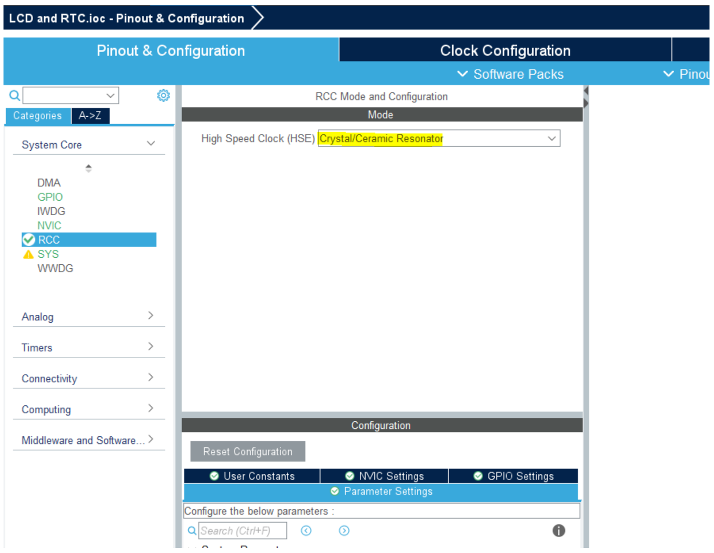
- 1 × Breadboard
- 1 × STM32F030F4P6
- 1× LCD 16x2
- 1× 10KΩ potentiometer
- 1x HCSR04 Ultrasonic Sensor
- Some Jumper wire

Procedure

Follow the circuit diagram shown in the image given below.



STM32F0 Pin Configuration:



LCD and RTC.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration

Software Packs | Pinou

Categories | A->Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- ✓ RCC
- ⚠ SYS
- WWDG

Analog

Timers

Connectivity

Computing

Middleware and Software...

RCC Mode and Configuration

Mode

High Speed Clock (HSE) Crystal/Ceramic Resonator

Configuration

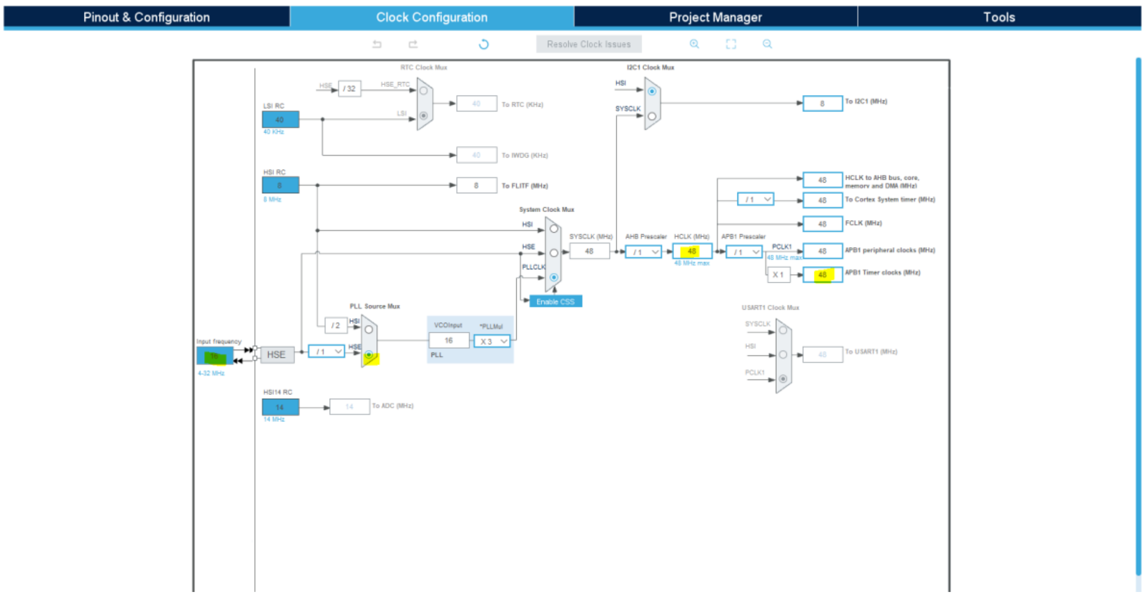
Reset Configuration

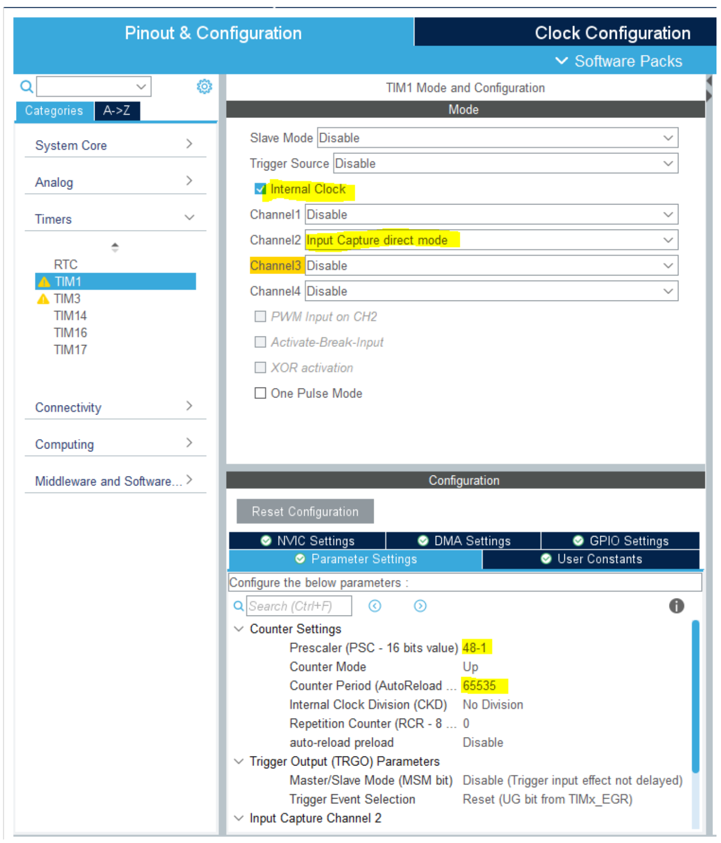
✓ User Constants | ✓ NVIC Settings | ✓ GPIO Settings

✓ Parameter Settings

Configure the below parameters :

Search (Ctrl+F)





TIM1 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

☑ Internal Clock

Channel1 Disable

Channel2 Input Capture direct mode

Channel3 Disable

Channel4 Disable

☐ *PWM Input on CH2*

☐ *Activate-Break-Input*

☐ *XOR activation*

☐ One Pulse Mode

Configuration

Configure the below parameters :

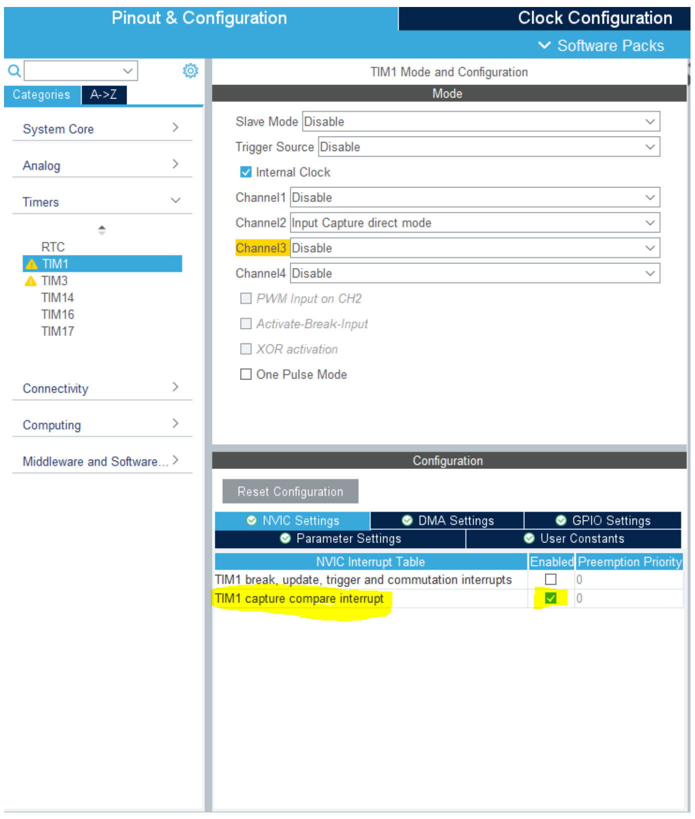
Counter Settings

Prescaler (PSC - 16 bits value)	48-1
Counter Mode	Up
Counter Period (AutoReload ...	65535
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 ...	0
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

Input Capture Channel 2



TIM1 Mode and Configuration

Mode

Slave Mode Disable

Trigger Source Disable

☑ Internal Clock

Channel1 Disable

Channel2 Input Capture direct mode

Channel3 Disable

Channel4 Disable

☐ *PWM Input on CH2*

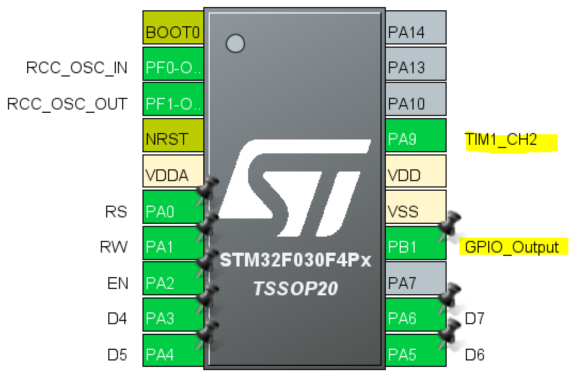
☐ *Activate-Break-Input*

☐ *XOR activation*

☐ One Pulse Mode

Configuration

NVIC Interrupt Table	Enabled	Preemption Priority
TIM1 break, update, trigger and commutation interrupts	☐	0
TIM1 capture compare interrupt	☑	0



Pin PA0 To PA6 set as Output for LCD

Code

```
#include "main.h"

#include "LCD1602.h"
TIM_HandleTypeDef htim1;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);

void delay_us (uint16_t time)
{
	__HAL_TIM_SET_COUNTER(&htim1, 0);
	while (__HAL_TIM_GET_COUNTER (&htim1) < time);
}

uint32_t IC_Val1 = 0;
uint32_t IC_Val2 = 0;
uint32_t Difference = 0;
uint8_t Is_First_Captured = 0;  // is the first value captured ?
uint8_t Distance  = 0;

#define TRIG_PIN GPIO_PIN_1
#define TRIG_PORT GPIOB

// Let's write the callback function

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
	if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)  // if the interrupt source is 
channel2
	{
		if (Is_First_Captured==0) // if the first value is not captured
		{
			IC_Val1 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2); // read the 
first value
			Is_First_Captured = 1;  // set the first captured as true
			// Now change the polarity to falling edge
			__HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_2, 
TIM_INPUTCHANNELPOLARITY_FALLING);
		}

		else if (Is_First_Captured==1)   // if the first is already captured
		{	// read second value
			IC_Val2 = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2);
		__HAL_TIM_SET_COUNTER(htim, 0);  // reset the counter

			if (IC_Val2 > IC_Val1)
			{
				Difference = IC_Val2-IC_Val1;
			}

			else if (IC_Val1 > IC_Val2)
			{
				Difference = (0xffff - IC_Val1) + IC_Val2;
			}
```

```
                  /*

                    Distance = (Time * Speed)/2;
                    speed of sound = 340 m/s = 0.034 cm/us
                    Distance = (Time[us] * Speed[cm/us])/2 = distance[us]

                    */

                    Distance = Difference * .034/2;
                    Is_First_Captured = 0; // set it back to false

                    // set polarity to rising edge
                    __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_2,
TIM_INPUTCHANNELPOLARITY_RISING);
                    __HAL_TIM_DISABLE_IT(&htim1, TIM_IT_CC2);
             }
      }
}

void HCSR04_Read (void)
{
      HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET);  // pull the TRIG pin HIGH
      delay_us(10);  // wait for 10 us
      HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);  // pull the TRIG pin low

      __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_CC2);
}


int main(void)
{
  HAL_Init();

  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM1_Init();

  HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_2);

  HAL_TIM_Base_Start(&htim1);  // Timer On and init this line
  lcd_init();                 //LCD init

  lcd_xy(0, 4);   // set curser postion
  LCD_String("Distance");  // Print String
  while (1)
  {
        HCSR04_Read ();
        lcd_xy(1, 7);
        LCD_intValue(Distance);
        HAL_Delay(500);
  }

}
```

Output: